Learning a Behavioral Repertoire from Demonstrations

Niels Justesen, Miguel González-Duque, Daniel Cabarcas, Jean-Baptiste Mouret, Sebastian Risi

Learning a Behavioral Repertoire from Demonstrations

Niels Justesen[1*], Miguel González-Duque[1*], Daniel Cabarcas[2], Jean-Baptiste Mouret[3], Sebastian Risi[1]
[1]*IT University of Copenhagen*,
[2]*Escuela de Matemáticas, Universidad Nacional de Colombia*,
[3]*Inria, CNRS, Universite de Lorraine*
[*]*These authors contributed equally*

***Abstract*—Imitation Learning (IL) is a machine learning approach to learn a policy from a set of demonstrations. IL can be useful to kick-start learning before applying reinforcement learning (RL) but it can also be useful on its own, e.g. to learn to imitate human players in video games. Despite the success of systems that use IL and RL, how such systems can adapt in-between game rounds is a neglected area of study but an important aspect of many strategy games. In this paper, we present a new approach called *Behavioral Repertoire Imitation Learning* (BRIL) that learns a *repertoire of behaviors* from a set of demonstrations by augmenting the state-action pairs with behavioral descriptions. The outcome of this approach is a single neural network policy conditioned on a behavior description that can be precisely modulated. We apply this approach to train a policy on 7,777 human demonstrations for the build-order planning task in StarCraft II. Dimensionality reduction is applied to construct a low-dimensional behavioral space from a high-dimensional description of the army unit composition of each human replay. The results demonstrate that the learned policy can be effectively manipulated to express distinct behaviors. Additionally, by applying the UCB1 algorithm, the policy can adapt its behavior – in-between games – to reach a performance beyond that of the traditional IL baseline approach.**

***Index Terms*—StarCraft II, imitation learning, build-order planning, online adaptation**

I. Introduction

Deep reinforcement learning has shown impressive results, especially for board games [1] and video games [2]. In some games it is sufficient to learn one strong policy, e.g. by finding an approximate global optimum. In other games, especially those with non-transitive strategies where there is no single optimal policy, it can be an advantage to learn a repertoire of strong and diverse policies. A policy repertoire also allows for inter-game adaption, i.e. adjusting the strategy in-between game rounds, which is an important strategic aspect in strategy games despite being neglected in game AI research [3]. The Real-time Strategy (RTS) game StarCraft (Blizzard Entertainment, 1998) is a good example of a game that requires inter-game adaptation and is has been a testbed for game AI research [4; 5]. The recent StarCraft bot *AlphaStar* learned a repertoire of policies through the so-called *AlphaLeague* wherein diversity was fostered through a massively parallel reinforcement learning system that rewarded different agents differently [6]. Despite having a repertoire of different policies, AlphaStar does not adapt in-between game rounds. In the AlphaLeague, diverse strategies were discovered with reinforcement learning, while Imitation Learning (IL) from human replays were shown necessary to bootstrap the learning process.

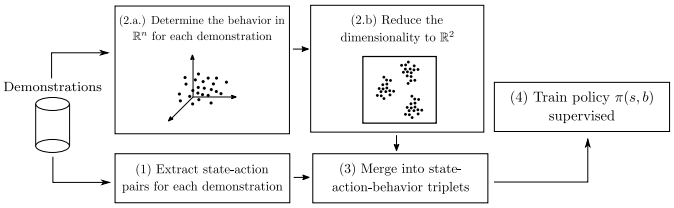

Fig. 1. **Behavioral Repertoire Imitation Learning (BRIL)** trains a policy $\pi(s, b)$ supervised on a data set of state-action pairs augmented with behavioral descriptors for each demonstration. When deployed, a system can adapt its behavior by modulating b (Sec. III-C). Optionally, high-dimensional behavioral spaces can be reduced using dimensionality reduction, as low-dimensional behavioral descriptions allow for better visualization and faster adaption.

In this paper, we present a new method called *Behavioral Repertoire Imitation Learning* (BRIL) that demonstrates how IL alone can discover strong and diverse policies suitable for inter-game adaptation. BRIL learns a single policy network that receives a behavioral description as an additional input vector. We show that a policy learned with BRIL can express a large space of behaviors and that it enables effective inter-game adaptation for a simple StarCraft bot. To our knowledge, no prior imitation learning techniques have demonstrated to facilitate inter-game adaptation.

BRIL consists of a multi-step process (Fig. 1) wherein the experimenter: (1) extracts state-action pairs (similarly to many IL approaches), (2) applies expert domain knowledge to design a set of behavioral dimensions to form a behavioral space (inspired by Quality-Diversity (QD) algorithms [7; 8]) and determines the behavioral description (coordinates in the space) for each demonstration, (3) merges the data to form a dataset of state-action-behavior triplets, and (4) trains a model to predict actions from state-behavior pairs through supervised learning. When deployed, the model can act as a policy and the behavior of the model can be manipulated by changing its behavioral input features.

BRIL is tested on the build-order planning problem in StarCraft II [9; 10; 11; 12], in which a high-level policy controls the build-order decisions of a bot that use scripted modules for low-level tasks. Real-Time Strategy (RTS) games, such as the StarCraft saga, are among the hardest games for

algorithms to learn as they contain a myriad of problems, such as dealing with imperfect information, adversarial real-time planning, sparse rewards, or huge state and action spaces [13]. Several algorithms and bots have been built [4; 14] to compete in AI tournaments such as the AIIDE StarCraft AI Competition[1], the CIG StarCraft RTS AI Competition[2], and the Student StarCraft AI Competition[3]. While our StarCraft II bot is much simpler and weaker than many other bots (including AlphaStar), our results demonstrate how adaptation using a behavioral repertoire can improve a bot and potentially allow it to become robust to exploitation which is a great concern for many AI systems.

II. Background

A. Imitation Learning

While Reinforcement Learning (RL) deals with learning a mapping (a policy) between states and actions by interacting with an environment, in Imitation Learning (IL) a policy is learned from demonstrations. Methods based on IL, also known as Learning from Demonstration (LfD), have shown promise in the field of robotics [15; 16; 17; 18; 19] and games [6; 10; 20; 21; 22; 23; 24].

IL is a form of supervised learning, in which the goal is to learn a policy $\pi(s)$, mapping a state $s \in S$ to a probability distribution over possible actions. In contrast to an RL task, the agent is presented with a dataset D of demonstrations. A demonstration $d_j \in D$ consists of k_j sequential state-action pairs, where the action was taken in the state by some policy. While not a general requirement, in this paper a demonstration corresponds to an episode, i.e. starting from an initial state and ending in a terminal state.

Generative Adversarial Imitation Learning (GAIL) uses a Generative Adversarial Network (GAN) architecture wherein the generator is a policy that produces trajectories (without access to rewards) and the discriminator has to distinguish between the generated trajectories and trajectories from a set of demonstrations [25]. Two extensions of GAIL learn a latent space of the demonstrations [26; 27], which results in a conditioned policy similar to our approach. While the BRIL approach introduced here requires manually designed behavioral dimensions, it also gives the user more control over the learned policy; different behavioral spaces can be beneficial for different purposes. A latent space also does not explicitly bare meaning, in contrast to a manually defined behavioral space. Additionally, our approach learns a low-dimensional behavioral space that is suitable for fast adaptation and human inspection.

B. Quality Diversity & Behavioral Repertoires

Traditional optimization algorithms aim at finding the optimal solution to a problem. Quality Diversity (QD) algorithms, on the other hand, attempt to find a set of high-performing but behavioral diverse solutions [7]. QD algorithms usually rely on evolutionary algorithms, such as Novelty Search with Local Competition (NSLC) [28] or MAP-Elites [8]. NSLC is a population-based multi-objective evolutionary algorithm with a novelty objective that encourages diversity and a local competition objective that measures an individual's ability to outperform similar individuals in the population. Individuals are added to an archive throughout the optimization process if they are significantly more novel than previously explored behaviors. MAP-Elites does not maintain a population throughout the evolutionary run, only an archive divided into cells that reflect the concept of behavioral niches in a pre-defined behavioral space. For example, different cells in the map correspond to different walking gaits for a hexapod robot [29]. Both NSLC and MAP-Elites results in an archive of diverse and high-performing solutions. The pressure toward diversity in QD algorithms can help the optimization process escape local optima [30], while the diverse set of solutions also allows for online adaption by switching intelligently between these [31]. We will describe variations of such an adaption procedure in the next section.

QD is related to the general idea of learning behavioral repertoires. Where QD algorithms optimize towards a single quality objective by simultaneously searching for diversity, a behavioral repertoire can consist of solutions optimized towards different objectives as in the Transferability-based Behavioral Repertoire Evolution algorithm (TBR-Evolution) [29].

C. Bandit Algorithms & Bayesian Optimization

Given either a discrete set or a continuous distribution of options, we can intelligently decide which options to select to maximize the expected total return over several trials. To do this, we consider the discrete case as a k-armed bandit problem and the continuous case as a Bayesian optimization problem. In the continuous case, the problem can also be simplified to a k-armed bandit problem, simply by picking k options from the continuous space of options.

The goal of a k-armed bandit problem is to maximize the total expected return after some number of trials by iteratively selecting one of k arms/options, each representing a fixed distribution of returns [32]. To solve this problem, one must balance exploitation (leveraging an option that has rendered high returns in the past) and exploration (trying options to gain a better estimation of their expected value). A bandit algorithm is a general solution to k-armed bandit problems. One of the most popular of these is the Upper Confidence Bound 1 (UCB1) algorithm [33] that first tries each arm once and then always selects the option that at each step maximizes:

$$\overline{X}_j + C\sqrt{\frac{2\ln t}{n_j}}, \tag{1}$$

where $\overline{X}_j$ is the mean return when selecting option j after t steps, n_j is the number of times option j has been selected, and C is a constant that determines the level of exploration.

[1] http://www.cs.mun.ca/~dchurchill/starcraftaicomp/

[2] http://cilab.sejong.ac.kr/sc_competition/

[3] http://sscaitournament.com/

This k-armed bandit approach can be considered as the discrete case of the more general approach of Bayesian optimization (BO), in which a continuous black-box objective function is optimized. BO starts with a prior distribution over objective functions, which is then updated based on queries to the black-box function using Bayes' theorem. The Intelligent Trial-and-Error algorithm (IT&E) uses BO for robot adaptation to deal with changes in the environment by intelligently searching in the continuous behavioral space of policies found by MAP-Elites [31]. In their approach, the fitness of all solutions in the behavioral space is used to construct a prior distribution of the fitness, which is also called a behavior-performance map. A Bayesian optimizer is then used to sample a point in the map, record the observed performance, and compute a posterior distribution of the fitness. This process is continued until a satisfying solution is found. IT&E could also be applied as an adaptation procedure to the policy found by BRIL, by creating a prior distribution based on some quality information of the demonstrations, e.g. player rating or win-rate. A Bayesian optimizer is then used to optimize the behavioral feature input. In this paper we will, however, simplify the adaption process to a discrete k-armed bandit problem with k manually selected behavioral features and leave the use of Bayesian optimization techniques for future work.

D. Dimensionality Reduction

In this paper, the dimensionality of the behavior space is reduced using the t-distributed Stochastic Neighbor Embedding (t-SNE) by [34]. In Stochastic Neighbor Embedding (SNE, a precursor to t-SNE), a graph is embedded by minimizing the distance between two probability distributions measured with the Kullback-Leibler divergence. The first of these probability distributions reflect the similarity between the high-dimensional points, and the second one measures the similarity between the embedded, low dimensional points. The high-dimensional probability is fixed, while the embedding is iteratively updated to minimize the distance between its probability distribution and the fixed one. t-SNE uses a Student t-distribution kernel for the embedding's probability, solving previously known flukes of SNE such as the crowding problem.

E. Universal Policies

In value-based RL, one typically learns a state value function $V_\pi(s)$ or a state-action value function $Q_\pi(s,a)$ for a policy π. Universal Value Function Approximators (UVFA) instead learn a joint distribution $V_\pi(s,g)$ or $Q_\pi(s,a,g)$ over all goals $g \in G$ [35]. UVFA can be learned using supervised learning from a training set of optimal values such as $V^*_g(s)$ or $Q^*_g(s,a)$, or it can be learned through RL by switching between goals both when generating trajectories and when computing gradients. Hindsight Experience Replay is an extension to UVFAs, which performs an additional gradient update with the goal being replaced by the terminal state; this modification can give further improvements when it is infeasible to reach the goals [36]. An extension to Generative Adversarial Imitation Learning (GAIL) augments each trajectory with a *context* [37], which specifies the agent's sub-goals that can be modulated at test-time.

In our approach, we are not considering goals, but rather behaviors, intending to learn a universal policy $\pi(s,b)$ over states $s \in S$ and behaviors $b \in B$ in a particular behavioral space. We are thus combining the QD approach of designing a behavioral space with the idea of learning a universal policy to express behaviors in this space.

III. Behavioral Repertoire Imitation Learning (BRIL)

This section describes two approaches to learning behavioral repertoires using IL. We first describe how a behavioral space can be formed from demonstrations. Then we introduce a naive IL approach that first clusters demonstrations based on their coordinates in the behavioral space, and then applies traditional IL on each cluster. Finally, BRIL is introduced, which learns a single policy augmented with a behavioral feature input rather than learning multiple policies for each behavioral cluster.

A. Behavioral Spaces from Demonstrations

A behavioral space consists of some behavioral dimensions that are typically determined by the experimenter. For example, in StarCraft and StarCraft II, behavioral dimensions can correspond to the ratio of each army unit produced throughout the game to express the strategic characteristics of the player. A behavioral space can require numerous dimensions to be able to express meaningful behavioral relationships between interesting solutions for a problem. Intuitively, if the problem is complex, more dimensions can give a finer granularity in the diversity of solutions. However, there is a trade-off between granularity and adaptation, as low-dimensional spaces are easier to search in. We thus propose the idea of first designing a high-dimensional behavioral space and then reducing the number of dimensions through dimensionality reduction techniques. This second step, however, is optional. In our preliminary experiments, it has been beneficial to reduce the space to two dimensions, as it allows for easy visualization of the data distribution and it also seems to be a good trade-off between granularity and adaptation speed. With one-dimensional behavioral spaces, we noticed that nearby solutions could be wildly different.

B. Imitation Learning on Behavioral Clusters

The naive IL approach for learning behavioral repertoires trains n policies on n behaviorally diverse subsets of the demonstrations. This idea is similar to the state-space clustering in Thurau et al. [22], but we cluster data points in a behavioral space instead. When a behavioral space is defined, each demonstration can be specified by a particular behavioral description (a coordinate in the $\mathbb{R}^n$ dimensional space), where afterwards a clustering algorithm can split the dataset into several subsets. Hereafter, traditional IL can be applied to each subset to learn one policy for each behavioral cluster. This

approach creates a discrete set of policies similar to current QD algorithms. However, it introduces a dilemma: if the clusters are small, there is a risk of overfitting to these reduced training sets. On the other hand, if the clusters are large but few, the granularity of behaviors is lost.

C. Learning Behavioral Repertoires

QD algorithms typically fill an archive with diverse and high-quality solutions, sometimes resulting in thousands of policies stored in a single run, which increases the storage requirements in training as well as in deployment. To reduce the storage requirement, one can decrease the size of the archive, with the trade-off of losing granularity in the behavioral space. The main approach introduced in this paper, called Behavioral Repertoire Imitation Learning (BRIL), solves these issues and reduces overfitting by employing a universal policy instead, in which a single policy is conditioned on a behavioral description. In contrast to QD algorithms, the goal of BRIL is neither to optimize quality nor diversity directly. Instead, BRIL attempts to imitate and express the diverse range of behaviors and the quality that exists in a given set of demonstrations. Additionally, BRIL produces a continuous space of policies which is potentially more expressive than a discrete set.

BRIL extends the traditional imitation learning setting through the following approach. First, the behavioral characteristics of each demonstration are determined. If the dimensionality of these descriptions is large, it can be useful to reduce the space as described in the earlier section. A training set of state-action-behavior triplets is then constructed, such that the behavior is equal to the behavioral description of the corresponding demonstration. Then, a policy $\pi(s, b)$ is trained in a supervised way on this dataset to map states and behaviors to actions. Following this approach, the training set is not reduced in contrast to IL on behavioral clusters.

When the trained policy is deployed, the behavioral feature input can be modulated to manipulate its behavior. The simplest approach is to fix the behavioral features throughout an episode, evaluate the episodic return, and then consider new behavioral features for the next episode. This approach should allow for episodic, or inter-game, adaptivity, which will be explored in our experiments. One could also manipulate the behavioral features during an episode e.g. by learning a meta-policy.

IV. Experiments

This section presents the experimental results of applying BRIL to the game of StarCraft. Policies are trained to control the build-order planning module of a relatively simple scripted StarCraft bot[4] that plays the Terran race. While the policy is trained off-line, our experiments attempt to optimize the playing strength of this bot online, in-between episodes/games, by manipulating its behavior.

[4]https://github.com/njustesen/sc2bot

A. Behavioral Feature Space

The behavioral space for a StarCraft build-order policy can be designed in many ways. Inspired by the AlphaStar League Strategy Map [6], the behavioral features are constructed from the army composition, such that the dimensions represent the ratios of each unit type. We achieve this by traversing all demonstrations in the data set, counting all the army unit creation events, and computing the relative ratios. Each demonstration thus has an n-dimensional behavioral feature description, where $n = 15$ is the number of army unit types for Terran.

To form a 2D behavioral space, which allows for easier online search and analysis, we apply t-Distributed Stochastic Neighbor Embedding (t-SNE). Fig. 2 visualizes the points of all the demonstrations in this 2D space and Fig. 2a shows four plots where the points are colored to show the ratios of Marines, Marauders, Hellions, and Siege Tanks that were produced during these games.

B. Clustering

For the baseline approach that applies IL to behavioral clusters, we use density-based spatial clustering of applications with noise (DBSCAN) [38] with $\epsilon = 0.02$ and a minimum number of samples per cluster of 30. These two parameters were found through manual experimentation to find the most meaningful data separation; however, the clustering is not perfect due to the many outliers. The clusters are visualized in Fig. 2b, with outliers shown in black.

C. Performance in StarCraft

We trained three groups of neural networks, all with three hidden layers and 256 hidden nodes per layer: (1) One baseline model trained on the whole dataset with no augmentation of behavioral features, (2) a BRIL model on the whole dataset with two extra input nodes for the behavioral features (i.e. the coordinates in Fig. 2b), and (3) several cluster baseline models trained only on demonstrations from their respective clusters without the augmented behavioral features. BRIL used UCB1 for adaptation with an exploration constant $C = 1$.

We applied these trained policy models as build-order modules in the scripted bot. It is important to note that this is a very simplistic bot with several flaws and limitations. Therefore the main goal in this paper is not to achieve human-level performance in StarCraft, but rather to test if BRIL allows us to do manipulate its behavior and enables online adaptation. The build-order module, here controlled by one of our policies, is queried with a state description and returns a build-order action, i.e. which building, research, or unit to produce next. The worker and building modules of the bot perform these actions accordingly, while assault, scout, and army modules control the army units. Importantly, policies we test act in a system that consists of both the bot, the opponent bot, and the game world. When we want to utilize our method for adaptation, we are thus not only adapting to the opponent but also the peculiarities of the bot itself.

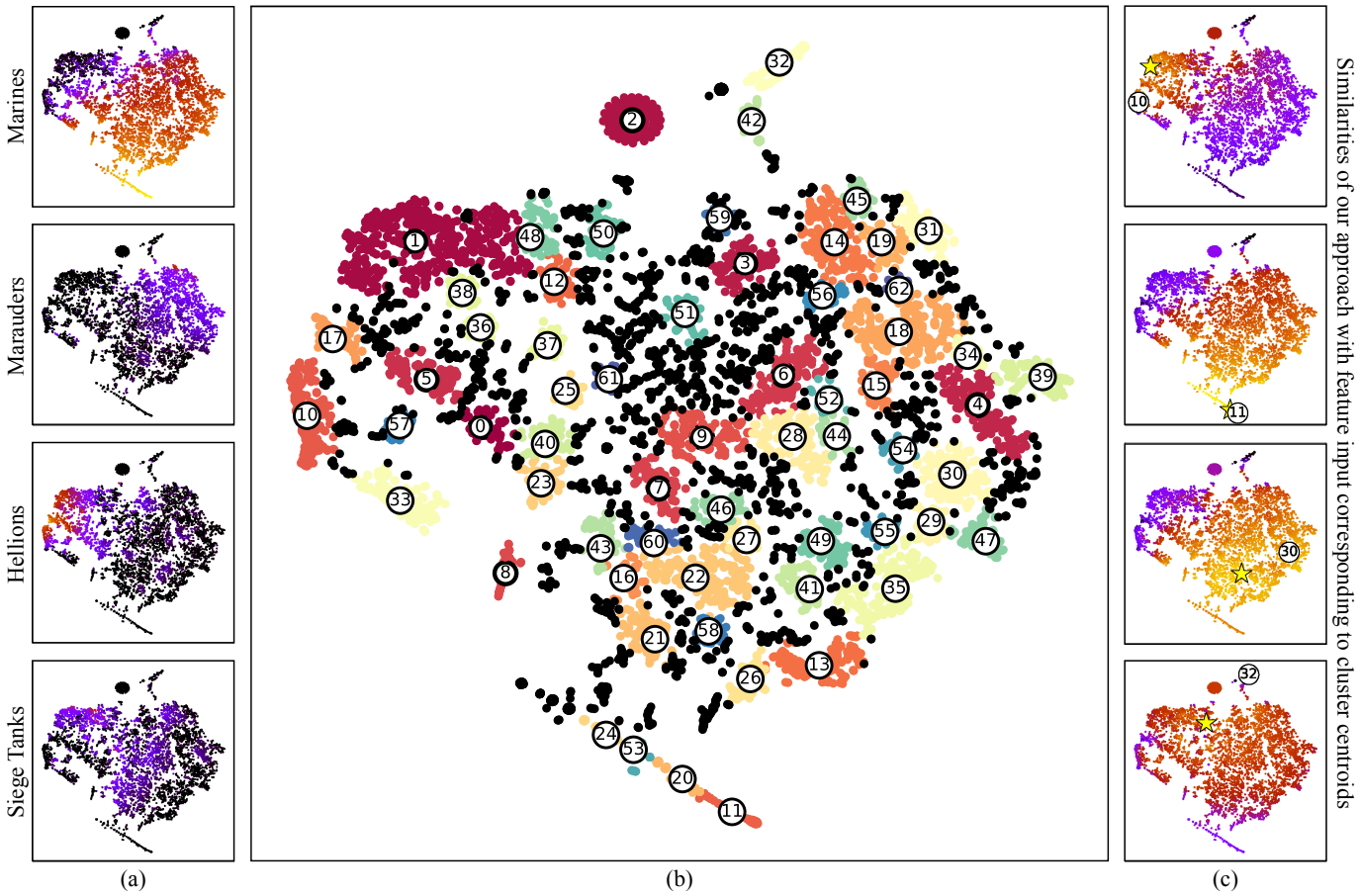

Fig. 2. **Visualizations of the 2D behavioral space of Terran army unit combinations in 7,777 Terran versus Zerg replays.** Each point represents a replay from the Terran player's perspective. The space was reduced using t-SNE. (a) The data points are illuminated (black is low and yellow is high) by the ratio of Marines, Marauders, Hellions, or Siege Tanks produced in each game. (b) 62 clusters found by DBSCAN. Cluster centroids are marked with a circle and the cluster number and outliers are black. The noticeable cluster 2 has no army units. (c) The similarity between the behaviors of the human players and our approach with four different feature inputs, corresponding to the coordinates of centroids of cluster 10, 11, 30, and 32. The behavior of our approach is averaged over 100 games against the easy Zerg bot and its nearest human behavior is marked with a star. The behavior of the learned policy can be efficiently manipulated to change its behavior. Additionally, we can control the behavior such that it resembles the behavior of a human demonstration.

TABLE I

RESULTS IN STARCRAFT USING IMITATION LEARNING (IL) ON THE WHOLE TRAINING SET, IL ON INDIVIDUAL CLUSTERS (C10, C11, C30, AND C32), AND BEHAVIORAL REPERTOIRE IMITATION LEARNING (BRIL) WITH FIXED BEHAVIORAL FEATURES CORRESPONDING TO CENTROIDS IN C10, C11, C30, AND C32. ADDITIONALLY, RESULTS ARE SHOWN IN WHICH UCB1 SELECTS BETWEEN THE FOUR BEHAVIORAL FEATURES IN-BETWEEN GAMES. EACH VARIANT PLAYED 100 GAMES AGAINST THE EASY ZERG BOT. THE NEAREST DEMONSTRATION IN THE ENTIRE DATASET WAS FOUND BASED ON THE BOT'S MEAN BEHAVIOR (NORMALIZED ARMY UNIT COMBINATION) AND THE DISTANCE TO EACH CLUSTER CENTROID ARE SHOWN. THE RESULTS DEMONSTRATE THAT BY USING CERTAIN BEHAVIORAL FEATURES FOR THIS PROBLEM, THE BRIL POLICY CAN OUTPERFORM THE TRADITIONAL IL POLICY AS WELL AS IL ON BEHAVIORAL CLUSTERS.

		Distance to cluster centroid				Combat units produced				
Method	Wins	C10	C11	C30	C32	Marines	Marauders	Hellions	S. Tanks	Reapers
IL	41/100	0.58	0.22	0.39	0.75	44.1 ± 50.5	0.7 ± 3.2	2.6 ± 7.6	1.7 ± 6.5	0.3 ± 1.1
IL (C10)	3/100	**0.05**	0.76	0.81	0.71	1.1 ± 2.3	0.1 ± 0.3	**3.11 ± 6.1**	**0.1 ± 0.4**	0.1 ± 0.33
IL (C11)	7/100	0.74	**0.00**	0.52	0.96	18.8 ± 38.4	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
IL (C30)	18/100	0.76	0.21	**0.31**	0.79	**43.5 ± 62.6**	**0.9 ± 5.4**	0.2 ± 1.3	0.0 ± 0.2	0.2 ± 0.8
IL (C32)	0/100	0.71	0.94	0.57	**0.04**	0.1 ± 0.2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	**9.9 ± 18.5**
BRIL (C10)	27/100	**0.21**	0.85	0.81	0.60	2.4 ± 4.9	0.0 ± 0.0	**14.6 ± 18.9**	4.0 ± 5.2	0.2 ± 0.6
BRIL (C11)	**76/100**	0.70	**0.05**	0.53	0.95	**81.4 ± 50.1**	0.0 ± 0.1	0.2 ± 0.1	0.9 ± 2.4	0.3 ± 0.6
BRIL (C30)	47/100	0.60	0.31	**0.29**	0.65	41.6 ± 36.4	**2.4 ± 6.7**	0.7 ± 2.5	4.1 ± 7.8	0.5 ± 1.2
BRIL (C32)	16/100	0.42	0.72	0.53	**0.36**	7.1 ± 11.4	1.7 ± 6.5	3.2 ± 8.3	**6.7 ± 9.7**	**0.8 ± 1.5**
		Wins for each option				Combat units produced				
Method	Win	C10	C11	C30	C32	Marines	Marauders	Hellions	Siege Tanks	Reapers
BRIL (UCB1)	61/100	5/14	**47/59**	8/18	1/9	52.1 ± 47.7	0.5 ± 3.2	4.3 ± 12.5	2.5 ± 6.2	0.3 ± 1.3

We will first focus on the results of the traditional IL approach. Table I shows the number of wins in 100 games on the two-player map CatalystLE as well as the corresponding average behaviors (i.e. the army unit ratios). Our bot played as Terran against the built-in Easy Zerg bot. The traditional IL approach won 41/100 games. IL on behavioral clusters showed very poor performance with a maximum of 18/100 wins by the model trained on C30. Besides the number of wins, we compute the nearest demonstration in the entire data set from the average behavior and use it as an estimate of the policy's position in the 2D behavioral space. From the estimated point, we calculate the distance to each of the four cluster centroids. These centroids were selected because they represent strategies that are far from each other in the behavioral space. This analysis revealed that the policies trained on behavioral clusters express behaviors close to the clusters they were trained on (note the distances to the cluster centroids in Table I). We hypothesize that the poor win rates of this naive approach are due to their training sets being too small such that the policies do not generalize to many of the states explored in the test environment.

Table I also shows the results for BRIL with the coordinates of the four cluster centroids as behavioral features. BRIL (C11) achieves a win rate of 76/100. These results demonstrate that, for some particular environments, the model can be tuned to achieve a higher performance than a policy found by traditional IL. Analyzing the behavior of the bot with the behavioral features of C11 reveals that it performs an all-in Marine push, similarly to the behavior of the demonstrations in C11 (notice the position of C11 on Fig. 2.b and the illumination of Marines on Fig. 2.a). With the behavioral features of C30, the approach reached a slightly higher win rate than traditional IL (47 vs. 41). We also notice that for both BRIL and IL on behavioral clusters, the average expressed behavior is closest to the cluster centroid that it was modulated to behave as, among the four clusters we selected. The results show that the behavior of the learned BRIL policy can be successfully controlled. However, the distances are on average larger than for IL on behavioral clusters.

D. Army Compositions In-game

Replays in each of the clusters we examined in the bandit problem (that is, C10, C11, C30 and C32) exhibit a particular army composition in the game. Cluster 11, for example, shows a strategy composed almost purely of Marines (see Fig. 2). Fig. 3 shows screenshots of typical army compositions produced by the BRIL policy with the four different behavioral features.

E. Online Adaptation

The final test aims to verify that we can indeed use BRIL for online adaptation. We apply the UCB1 algorithm to select behavioral features from the discrete set of four options: {C10, C11, C30, C32} (i.e. the two-dimensional feature descriptions of these cluster centroids). This approach enables the algorithm to switch between behavioral features in-between games based on the return of the previous one, which is 1 for a win and 0 otherwise. The adaptive approach achieves 61/100 wins by identifying the behavior of C11 as the best option. Not surprisingly, the win rate is lower than when having the behavioral features of C11 fixed, while it outperforms traditional IL.

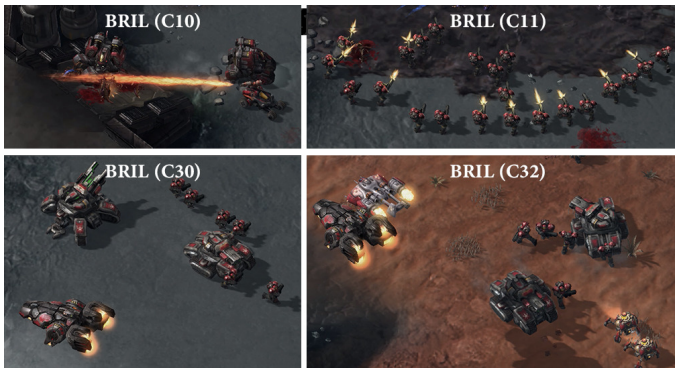

Fig. 3. Typical army compositions produced by our trained BRIL policy with behavioral features corresponding to the centroids of cluster 10, 11, 30 and 32. BRIL (C10) executes early timing pushes with Hellions and Cyclones, BRIL (C11) is aggressive with Marines only, BRIL (C30) creates mixed armies with many Marines and Siege Tanks, and BRIL (C32) also creates mixed armies but with less Marines and more Widow Mines.

V. Discussion

We proposed two new IL methods in this paper, one which learns a policy that is trained on only one behavioral cluster of data points, and one which learns a single modifiable policy on the whole dataset. Our results suggest that policies trained on small behavioral clusters overfit and are thus unable to generalize beyond the states available in the cluster. This drawback might be solved with fewer and larger clusters at the cost of losing granularity in the repertoire of policies. If data is abundant, this approach may also work better while we still suspect the same overfitting would occur. BRIL, on the other hand, is simple to implement and results in a continuous distribution of policies by adjusting the behavioral features. Additionally, the results suggest that BRIL generalizes better, most likely because it learns from the whole training set. However, that generality potentially comes with the cost of higher divergence between the expected behavior (corresponding to the behavioral input features) and the resulting behavior when tested. While an important concern, a divergence is somewhat expected since our test environment is very different from that of the training set (different maps and opponents). It is also important to note, that one behavior that works well in one scenario (e.g. BRIL (C11) in our experiments) may not work well in another scenario (e.g. on another map or against a different opponent). Adaptation is thus key for BRIL to yield good results. In setups where adaptation is not possible, BRIL may not be the preferred method. This could e.g. be in StarCraft ladder games where you rarely face the same opponent twice.

GAIL can likely be used in a similar way as BRIL since it also learns a distribution of policies that can be modulated. With BRIL, we can control behavioral dimensions rather than

using a latent space that does not necessarily capture the behaviors that we want to express. Important future work could explore the effects on adaption using either a learned and a user-defined behavioral space. The purpose of this paper is not to show that BRIL is superior to GAIL but rather to demonstrate how such methods can be used to achieve inter-game adaption in strategy games.

Previous work showed how IL can kick-start learning before applying RL [6; 20]. With BRIL, one can easily form a population of diverse solutions instead of just one, which may be a promising approach for domains with a plethora of strategic choices like StarCraft. Promising future work could thus combine BRIL with ideas from AlphaStar to automatically form the initial population of policies used in the AlphaStar League.

Another aspect worth discussing is the use of dimensionality reduction. Although optional, this step was helpful in visualizing and navigating the different behaviors that were expressed in the original high-dimensional behavioral space. We would expect a cleaner separation of the behaviors if we were to apply clustering algorithms to the original high-dimensional space, at the cost of no longer being able to visualize. Even though t-SNE is considered by most as the state-of-the-art method for dimension reduction, several other techniques could be explored. For instance, certain datasets' structure can be recovered in low dimensional space using simpler algorithms such as Principal Component Analysis (PCA). Otherwise, other methods such as Isometric Feature Mapping (Isomap) by [39], Locally Linear Embedding (LLE) by [40] and Uniform Manifold Approximation and Projection (UMAP) [41] could be used. The use of these different techniques may impact the way the repertoire of behaviors is separated in the low dimensional space.

VI. Conclusions

We introduced a new method called Behavioral Repertoire Imitation Learning (BRIL). By labeling each demonstration $d \in D$ with a behavior descriptor confined within a predefined behavioral space, BRIL can learn a policy $\pi(s, b)$ over states $s \in S$ and behaviors $b \in B$. In our experiments, a low-dimensional representation of the behavioral space was obtained through dimensionality reduction. The results in this paper demonstrate that BRIL can learn a policy that, when deployed, can be manipulated by conditioning it with a behavioral feature input b, to express a wide variety of behaviors. Additionally, the observed behavior of the policy resembles the behavior characterized by b. Furthermore, a BRIL trained policy can be optimized online by searching for optimal behavioral features in a given setting. In our experiments, a policy trained with BRIL was optimized online beyond the performance reached by traditional IL, using UCB1 to select among a set of discrete behavioral features.

Acknowledgments

Niels Justesen was financially supported by the Elite Research travel grant from The Danish Ministry for Higher Education and Science. Miguel González-Duque received financial support from the Universidad Nacional de Colombia. Jean-Baptiste Mouret is supported by the European Research Council under Grant Agreement No. 637972 (project ResiBots). The work was supported by the Lifelong Learning Machines program from DARPA/MTO under Contract No. FA8750-18-C-0103.

References

[1] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] N. Justesen, M. S. Debus, and S. Risi, "When are we done with games?" in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–8.

[4] D. Churchill, M. Preuss, F. Richoux, G. Synnaeve, A. Uriarte, S. Ontanón, and M. Certickỳ, "Starcraft bots and competitions," 2016.

[5] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser *et al.*, "Starcraft II: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[6] O. Vinyals *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1724-z

[7] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.

[8] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[9] D. Churchill and M. Buro, "Build order optimization in Starcraft," in *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[10] N. Justesen and S. Risi, "Learning macromanagement in StarCraft from replays using deep learning," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2017, pp. 162–169.

[11] Z. Tang, D. Zhao, Y. Zhu, and P. Guo, "Reinforcement learning for build-order production in StarCraft II," in *2018 Eighth International Conference on Information Science and Technology (ICIST)*. IEEE, 2018, pp. 153–158.

[12] P. Sun, X. Sun, L. Han, J. Xiong, Q. Wang, B. Li, Y. Zheng, J. Liu, Y. Liu, H. Liu *et al.*, "TStarBots: Defeating the Cheating Level Built-in AI in StarCraft

II in the Full Game," *arXiv preprint arXiv:1809.07193*, 2018.

[13] M. Buro, "Real-time strategy games: A new ai research challenge," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 1534–1535.

[14] S. Ontanón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A survey of real-time strategy game ai research and competition in Starcraft," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 5, no. 4, pp. 293–311, 2013.

[15] P. Bakker and Y. Kuniyoshi, "Robot see, robot do: An overview of robot imitation," in *AISB96 Workshop on Learning in Robots and Animals*, 1996, pp. 3–11.

[16] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97. Citeseer, 1997, pp. 12–20.

[17] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[18] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[19] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[20] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484–489, 2016.

[21] S. F. Gudmundsson, P. Eisen, E. Poromaa, A. Nodet, S. Purmonen, B. Kozakowski, R. Meurling, and L. Cao, "Human-like playtesting with deep learning," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.

[22] C. Thurau, C. Bauckhage, and G. Sagerer, "Imitation learning at all levels of game-AI," in *Proceedings of the international conference on computer games, artificial intelligence, design and education*, vol. 5, 2004.

[23] B. Gorman and M. Humphrys, "Imitative learning of combat behaviours in first-person computer games," *Proceedings of CGAMES*, 2007.

[24] J. Harmer, L. Gisslén, J. del Val, H. Holst, J. Bergdahl, T. Olsson, K. Sjöö, and M. Nordin, "Imitation learning with concurrent actions in 3D games," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.

[25] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[26] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess, "Robust imitation of diverse behaviors," in *Advances in Neural Information Processing Systems*, 2017, pp. 5320–5329.

[27] Y. Li, J. Song, and S. Ermon, "InfoGAIL: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3812–3822.

[28] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 211–218.

[29] A. Cully and J.-B. Mouret, "Evolving a behavioral repertoire for a walking robot," *Evolutionary computation*, vol. 24, no. 1, pp. 59–88, 2016.

[30] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19, no. 2, pp. 189–223, 2011.

[31] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[33] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, pp. 235–256, 2002.

[34] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[35] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International conference on machine learning*, 2015, pp. 1312–1320.

[36] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.

[37] J. Merel, Y. Tassa, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess, "Learning human behaviors from motion capture by adversarial imitation," *arXiv preprint arXiv:1707.02201*, 2017.

[38] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *The Thirteenth National Conference on Artificial Intelligence*, 1996, pp. 1–8.

[39] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science Reports*, vol. 290, no. December, pp. 151–180, 2000.

[40] L. Saul and S. Roweis, "An introduction to locally linear embedding," *Journal of Machine Learning Research*, vol. 7, 01 2001.

[41] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.